SANDIA REPORT
SAND2000-3228
Unlimited Release
Printed January 2001

An Examination of System Architectures for Distributing Sensor Data Via Ethernet Networks

K. B. Pfeifer, R. W. Cernosek, Stephen J. Martin, R. L. Waldschmidt, and A. N. Rumpf

Sandia National Laboratories

SAND2000-3228
Unlimited Release
Printed January 2001

An Examination of System Architectures for Distributing Sensor Data Via Ethernet Networks

by

K. B. Pfeifer[a], R. W. Cernosek[a], Stephen J. Martin[a],

R. L. Waldschmidt[b], A. N. Rumpf[b]

Sandia National Laboratories

[a]Microsensor Research and Development Dept., 1744

[b]L & M Technologies Corporation

Abstract

Application of the World Wide Web (WWW) for the transfer of sensor data from remote locations to laboratories and offices is a largely ignored application of the WWW. We have investigated several architectures for this application including simple web server/client architectures and variations of this approach. In addition, we have evaluated several commercial approaches and other techniques that have been investigated and are in the literature. Finally, we have provided conclusions based on the results of our study offering suggestions about the advantages and disadvantages of each of the approaches studied.

Table of Contents

Table of Figures

An Examination of System Architectures for Distributing Sensor Data Via Ethernet Networks

by

K. B. Pfeifer[a], R. W. Cernosek[a], S. J. Martin[a],

R. L. Waldschmidt[b], A. N. Rumpf[b]

Sandia National Laboratories

[a]Microsensor Research and Development Dept., 1744

[b]L & M Technologies Corporation

INTRODUCTION

The advent and subsequent world conquest by the World Wide Web (WWW) is a

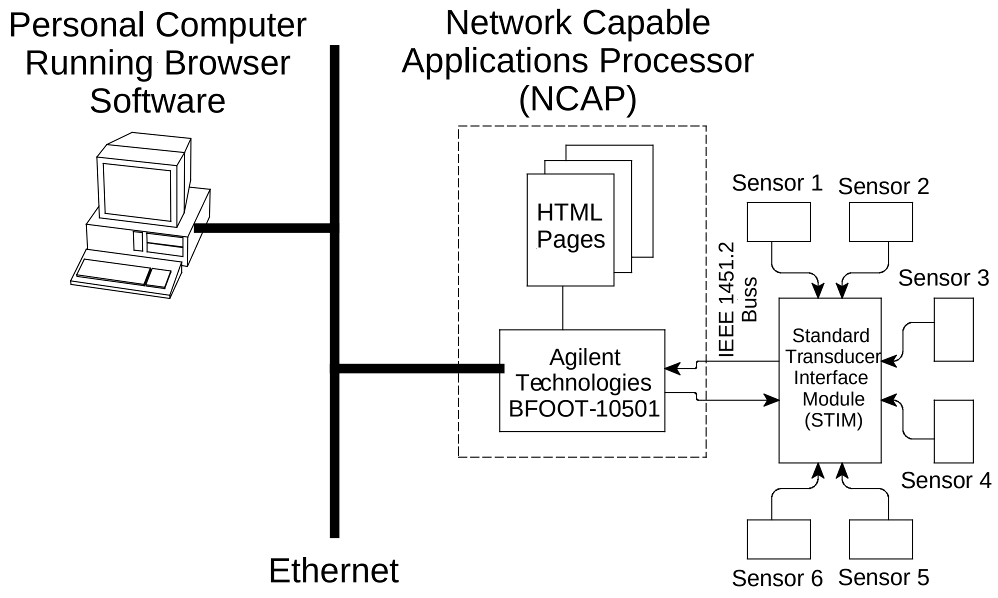

Personal Computer Running Browser Software
Network Capable Applications Processor (NCAP)
HTML Pages
Agilent Technologies BFOOT-10501
IEEE 1451.2 Buss
Standard Transducer Interface Module (STIM)
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Sensor 5
Sensor 6
Ethernet

Figure 1. *Diagram showing the architecture of the simple NCAP web server application. The computer is running a standard browser software package such as Netscape, and the NCAP functions as simple server with custom web pages downloaded to its flash memory.*

phenomenon heretofore unprecedented in the history of technology. In just a few short years, what was originally confined to government, industry, and academia has become common place in offices and homes with literally millions of web addresses vying for customers and viewers. One area that has been largely ignored by the web has been the useful application of web technology to the remote acquisition of data from various sensors. This is largely due to several factors including the cost of providing a server computer acting as the interface for various suites of sensors and the lack of security that accompanies the open format of the WWW.

One can make the argument that the availability of sensor data from large complex machinery on the web would allow for the streamlining of troubleshooting problems that occur in industrial applications where a company may possess expertise in a certain area that is remote from its production facilities. In the past, machinery operators would rely on telephone calls to the expert to help determine the solution to a problem or in the worst case require a plant visit by the expert. Neither of these solutions is adequate when there is a large quantity of sensor data to be considered or there is a reoccurring transient problem. The machinery operators must either describe the current condition of the machinery to the expert via telephone or fax him data to analyze. Neither approach is real-time or satisfactory when dealing with large complex systems.

Another area where simple remote access to sensor data is necessary is in the monitoring of health and operational characteristics of critical infrastructure such as highway bridges or pipelines. As was witnessed by the recent explosion and subsequent loss of life from a natural gas line failure near Carlsbad, NM, it is clear that the ability is needed to continuously monitor the health of a pipeline and transmit the data to a central location in a form that is both easily implemented and robust. Development of a system that can perform this type of measurement is critical, and it would be highly desirable if that system could simply interface to the WWW.

The technology currently exists to provide this type of data transfer from a remote plant to a company headquarters in many forms, from simple ASCII data transfers to complex systems that integrate security in the form of coding schemes, etc. However, some situations exist where the remoteness of the application and redundancy of the measurement, as in the pipeline example, preclude the required PC to PC connection that

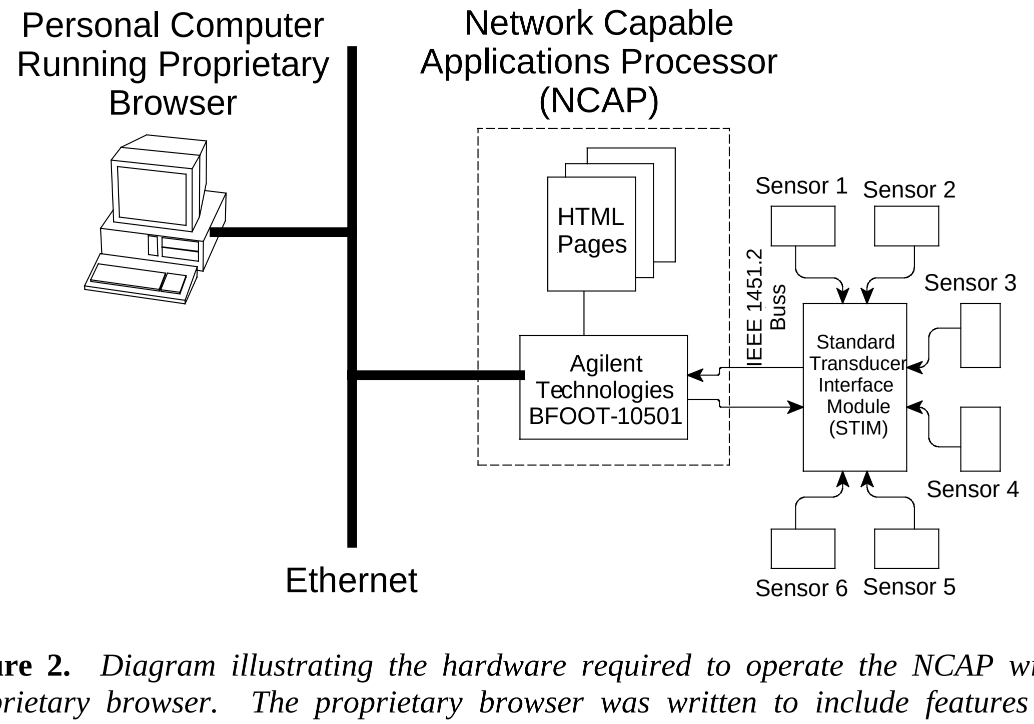

Personal Computer Running Proprietary Browser
Network Capable Applications Processor (NCAP)
HTML Pages
Agilent Technologies BFOOT-10501
IEEE 1451.2 Buss
Sensor 1
Sensor 2
Sensor 3
Sensor 4
Sensor 5
Sensor 6
Standard Transducer Interface Module (STIM)
Ethernet

Figure 2. *Diagram illustrating the hardware required to operate the NCAP with a proprietary browser. The proprietary browser was written to include features that cannot be accomplished using commercial browser software such as disk access and data storage.*

currently exists.

Recently, several companies have addressed this problem by producing hardware that operates as a web server and can distribute large quantities of data to web browsers. One

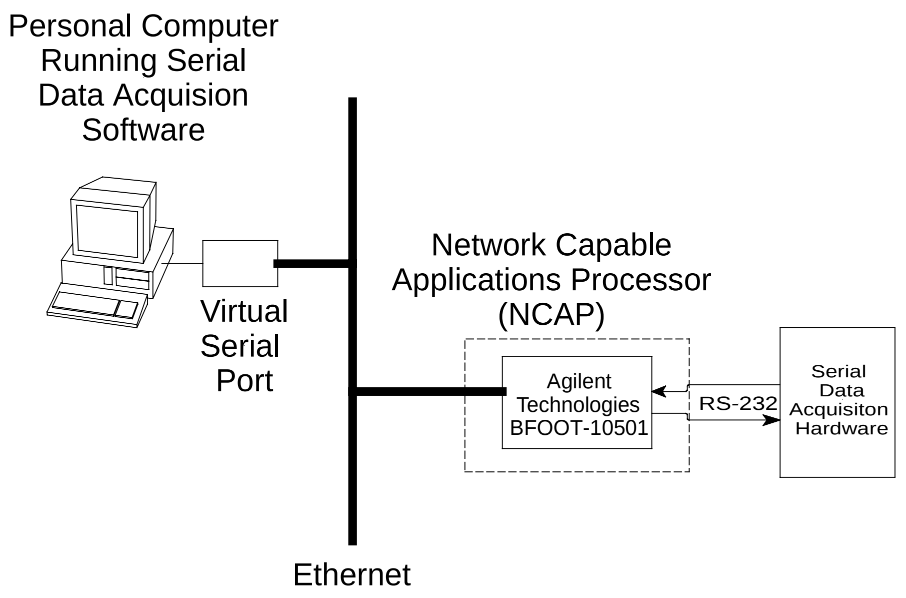

Personal Computer Running Serial Data Acquision Software
Virtual Serial Port
Network Capable Applications Processor (NCAP)
Agilent Technologies BFOOT-10501
RS-232
Serial Data Acquisiton Hardware
Ethernet

Figure 3. *Diagram illustrating the hardware required for the serial pass-through application. The PC is running a standard data acquisition software such as HP-Vee that communicates with the serial port. The computer is also running a software package such as Tactical Software's DialOut/IP that creates a virtual serial port and then sends the serial communication data via the ethernet to the NCAP. The NCAP then converts the ethernet signals back to RS-232 and sends then to an instrument with a common serial interface.*

such device is the Agilent Technologies BFOOT-10501[1]. We have demonstrated several architectures for placing data on the web using this device and will discuss their advantages and disadvantages for various applications. Additionally, we have investigated other techniques that utilize special hardware and software to interface sensor data to the WWW or another network. Those will be described briefly at the end this report.

EXPERIMENTAL

We have investigated three distinct types of web architectures including the simple web server/client system using a commercial browser software (Figure 1), developing a proprietary browser that has features such as the ability to save input data to disk (Figure 2), and using the web as a distributed serial interface (Figure 3). While it is relatively simple to program an HTML web page and implement a server client application, it is distinctly more complex to convert sensor data into a format that is easily web accessible. This function is provided using a circuit component called a Standard Transducer

Interface Module (STIM) which consists of hardware that digitizes the analog signal from a sensor and converts it to IEEE 1451.2 standard for transfer to the server module and eventually to the client browser.[2] The STIM is the system component that is completely dependent on the target application and cannot be produced in any generalized form that is sufficient for all or even most applications. The variable nature of each application requires that the sensor input be treated uniquely, and thus, for all but the simplest of applications, the STIM must become a custom design. The general form of a STIM is given in Figure 4 which illustrates the implementation of the IEEE 1451.2 format and the generalized hardware that can be associated with a STIM.

Server/Client Architecture

The first example that was implemented was the simple server/client application of Figure 1. This application employs a commercial STIM purchased from Telemonitor (TMI 1451.2-KC)[3]. This STIM has the capability to digitize six analog input channels (0-4.1 V), output two analog (0-4.1 V) channels, and output two digital channels for control applications. Our application consisted of measuring STIM circuit board temperature and the relative humidity as a function of time using a Hycal IH-3602-C[4] humidity sensor. These data are shown in Figure 5, which illustrates the application of the NCAP as a simple web server. The site source code is written in HTML (see Appendix A) and forms four frames in which four separate web pages are inserted. The first frame displays the date, time, and the Sandia National Laboratories symbols. The second frame includes links to pages that display temperature, humidity, atmospheric pressure, and room oxygen concentration. The third frame is a placeholder and in this example contains three photos, but in a more elaborate page would have links to other places or data acquisition. The final frame contains a JAVA applet that forms a strip chart record of the data and can be configured to measure up to four channels and display the data as a function of time.

This type of data display is useful for applications where current conditions are important but only measured infrequently. However, this type of data acquisition architecture has several severe drawbacks. First, commercial browser software such as *Netscape*[5] has built-in security features that preclude the web site from sending data to the remote computer's disk via *JavaScript* or *JAVA* applets for storage without permission. This is to protect the web surfing public from being infected with a virus by merely visiting a web site. Currently, a browser asks the user if he wants to directly open the file to be downloaded or if he wants to save it to disk. This is typically followed by a warning message stating that by directly opening the file, a virus can be transmitted. Because of this built-in feature, it is not possible to have the NCAP place data from the STIM directly into a file. However, by custom designing a STIM with onboard mass storage, it should be possible to have the STIM log data continuously in real time and then have the browser download the file to the remote computer. Thus, a simple STIM without data logging capability and an NCAP accessed by the typical browser is not an adequate combination for long term sensor data acquisition and storage.

Proprietary Browser

The second architecture that was tested is shown in Figure 2 and illustrates the hardware and software that is required to operate the NCAP using a proprietary browser

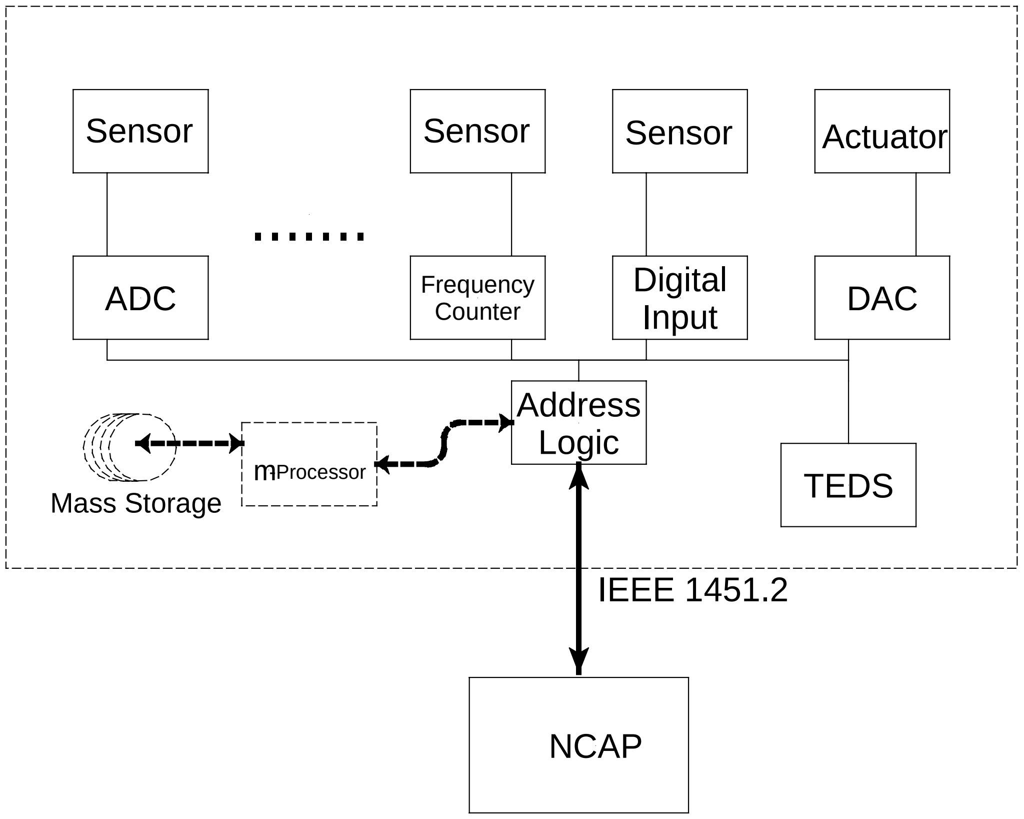


Figure 4. *Diagram of the STIM architecture showing the various types of sensors and their associated data-to-binary format conversion sections. In addition, the microprocessor and mass storage device allows for autonomy of the STIM to make data measurements that can be sent to the web and the client PC when called for by the client software. The data must be converted to IEEE 1451.2 standard prior to transfer to the NCAP. This requires the application of a Transducer Electronic Data Sheet (TEDS) section to the circuit. The information about the nature of the sensor data, scale factors to convert to real-world numbers and units are programmed in the TEDS.*

constructed from a software package such as *Visual Basic*.[6] This approach overcomes some of the limitations of operation since it allows direct access to the disk drives from the remote PC program but not from *JAVA* applets or *JAVAScript* code residing on the NCAP. Thus, data can be captured from the NCAP by sending a uniform resource locator (URL) to the NCAP via the web. This is done by the standard *Visual Basic* utility sending the following URL:

http://sale393.sandia.gov/bin/1452dot2/read?startChan=0&StopChan=5.

The section of the URL *sale393.sandia.gov* is the network address of our NCAP and corresponds to its IP address. The section */bin/1451dot2/* is a directory in the NCAP flash memory that contains the data extraction program for the specified channels; it returns the data along with the current elapsed time as measured by the NCAP. The section *read?startChan=0&stopChan=5* addresses a *JAVA* applet that measures the specified channels in sequence from *startChan* to *stopChan*. The result is a returned formatted data string that can be parsed by the *Visual Basic* program as illustrated in Appendix B. The data can then be saved or manipulated by the program as desired. This *Visual Basic* program also is shown in Appendix B.

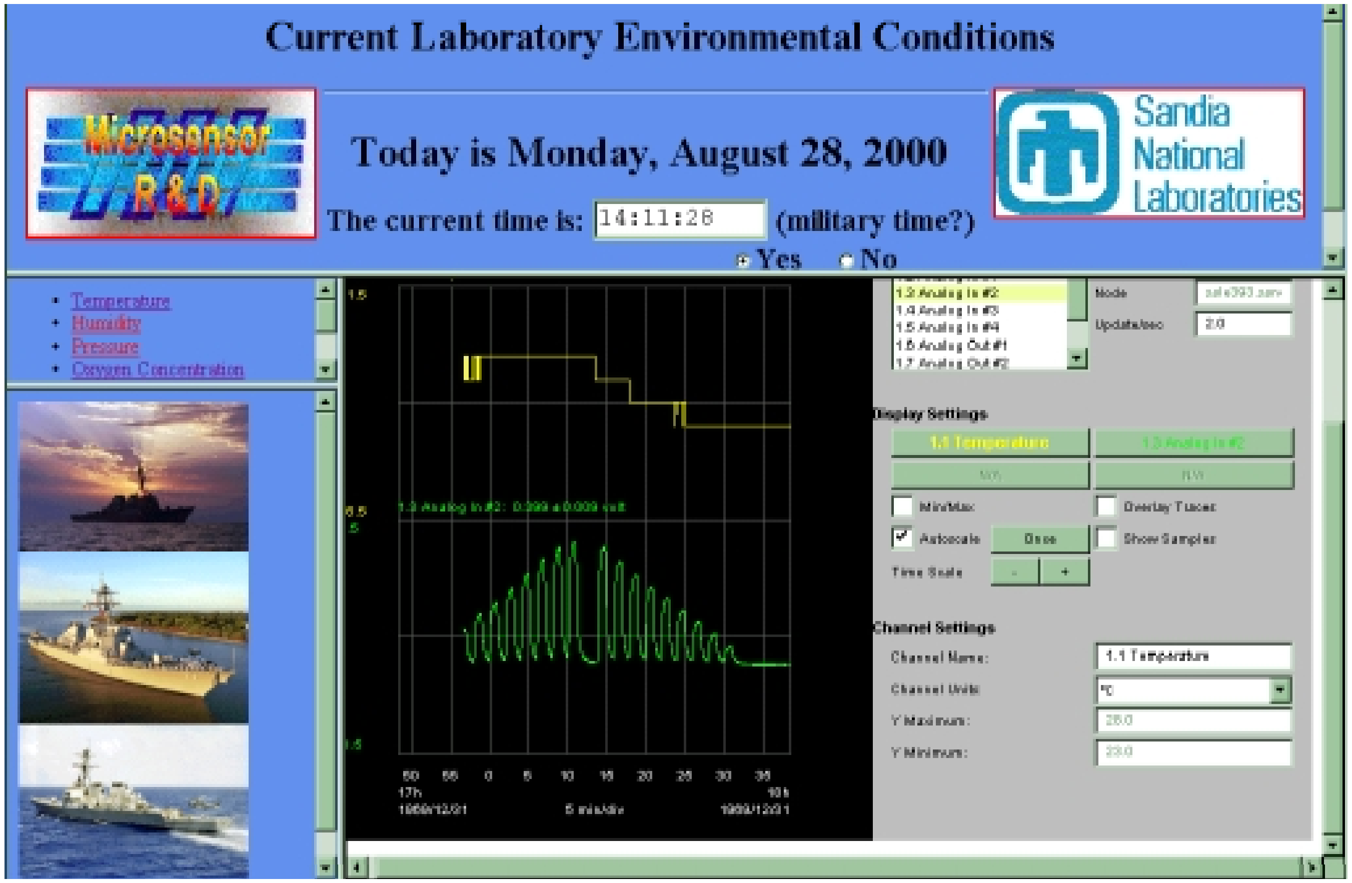
The same type of operation can be performed to write data to registers on the STIM. For example, the digital-to-analog converter (DAC) can be programmed remotely from *Netscape* or *Visual Basic* by sending the following URL:

http://sale393.sandia.gov/bin/11451dot2/write?chan.6="value"

where the command sections are the same as described previously. The section *write?chan.6="value"* is a *JAVA* applet that sends a value in volts to a DAC configured as channel 6, and the STIM then outputs a specified control voltage to operate some type of analog circuitry. We demonstrated this feature by sending a control voltage to a thermal electric (TE) cooler control circuit. The analog voltage did not drive the TE cooler directly, but used the analog signal to control a power transistor circuit that drove the TE cooler. The Telemonitor STIM also has digital outputs that can be controlled in a similar manner using the appropriate *write?* command structure.

Serial Pass-Through

The final architecture that we demonstrated is called a serial pass-through and is depicted in Figure 3. For this architecture, the NCAP simply performs the function of a serial to TCP/IP converter. The web becomes, in essence, a very long serial cable and is transparent to the software on the remote PC and the hardware being controlled. This architecture has the advantage of using existing software specially written to accomplish data acquisition along with commercial or custom hardware that has an RS-232 serial interface. The main limitations are (1) the rate of data capture is controlled by the operation of the WWW and cannot be as rapid as would be expected with a direct serial connection, and (2) serial communications via RS-232 are not addressable and, thus, the user is generally limited to a single measurement point.


Current Laboratory Environmental Conditions

Today is Monday, August 28, 2000

The current time is: 14:11:28 (military time?) [✓ Yes] No

- Temperature
- Humidity
- Pressure
- Oxygen Concentration

1.3 Analog In #2: 0.399 ±0.009 volt

1.3 Analog In #2
1.4 Analog In #3
1.5 Analog In #4
1.6 Analog Out #1
1.7 Analog Out #2

Node: sale393.sam
Update/sec: 2.0

Display Settings

[ ] Min/Max
[✓] Autoscale
[ ] Overlay Traces
[ ] Show Samples

Channel Settings

Channel Name: 1.1 Temperature
Channel Units: °C
Y Maximum: 28.0
Y Minimum: 23.0

Figure 5. *Displayed is a screen from Netscape illustrating the application of the NCAP as pure web server. The data shown in the graphs is for a commercial humidity sensor varied over 10% RH steps from 0 to 90% and then back to zero. The yellow trace is the temperature of the STIM hardware and varies only slightly over the period of the experiment.*

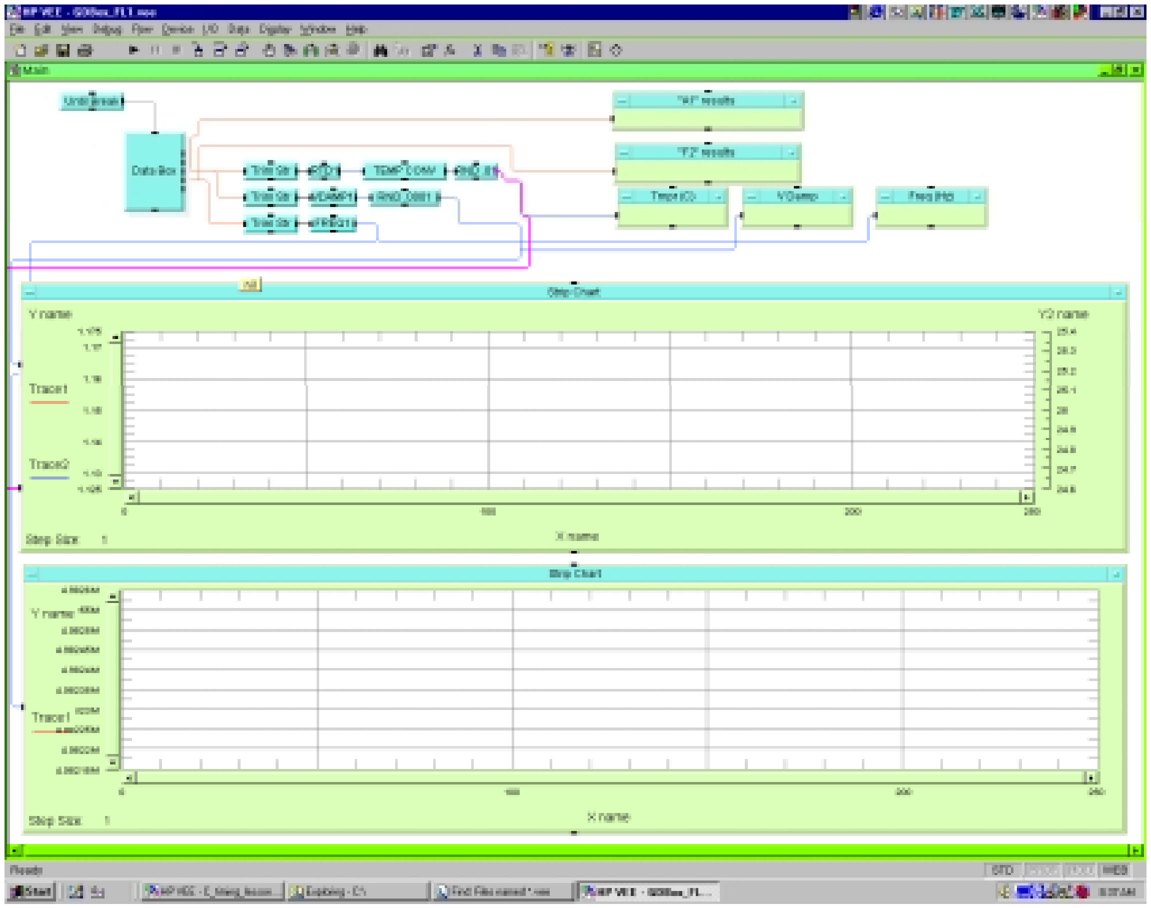


Figure 6. *Display of screen from the HP-Vee program running a quartz crystal-based viscosity monitor via RS-232. System was operated remotely via the web using an NCAP.*

As illustrated in Figure 3, the PC also must have a serial-to-TCP/IP converter. One way to accomplish this is to connect another NCAP to the serial-port of the controlling computer. The NCAP then converts the data to TCP/IP and sends it via the WWW to the second NCAP at the sensor site. This second NCAP then converts commands from TCP/IP to RS-232 and communicates with the serial hardware. In our application, we have chosen not to use an additional NCAP at the PC end of the system, but rather to install a software (virtual) serial port that accomplishes the same function without tying up a communication port or requiring the additional hardware.

We selected a product called *DialOut/IP 2.2.4* from Tactical Software[7] that, when installed, appears to any software package as a serial port. *DialOut/IP* converts the serial

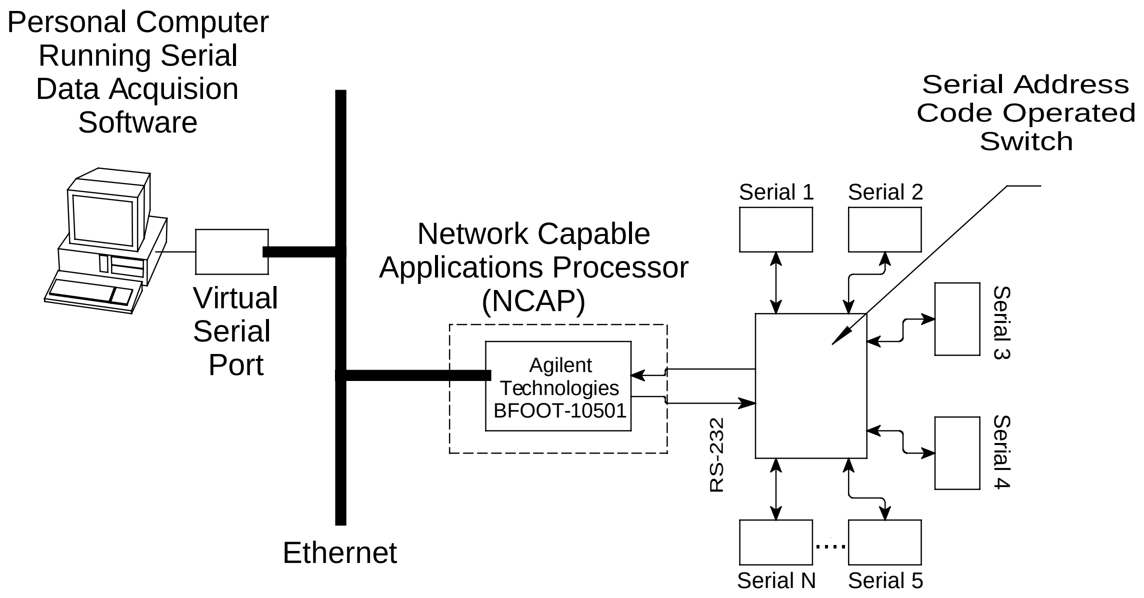

Personal Computer Running Serial Data Acquision Software
Virtual Serial Port
Ethernet
Network Capable Applications Processor (NCAP)
Agilent Technologies BFOOT-10501
RS-232
Serial Address Code Operated Switch
Serial 1
Serial 2
Serial 3
Serial 4
Serial 5
Serial N
....

Figure 7. *Diagram of the architecture using a serially addressed code-operated switch to allow access to multiple serial devices via the NCAP.*

data to TCP/IP and sends it via the computer's Ethernet connection and the WWW to the NCAP. The NCAP then converts the data to RS-232 and communicates with the devices.

The serial pass-through architecture has the advantage of allowing very sophisticated system data acquisition and control via the Internet in a form that can be encrypted for security. This is important not only for controlling access to information but also for data authentication. Additionally, this technique allows for continuous remote access to long term test equipment.

We demonstrated the serial pass-through architecture by connecting a computer running *HP-Vee* to a proprietary Sandia-developed sensor system that measures liquid viscosity. This system had previously been field tested by assembling all the equipment at the remote site and using a laptop computer running *HP-Vee* to acquire the data via a direct serial connection. The computer had to be located within the maximum range of RS-232 communications <<1 km. With this system, the computer running *HP-Vee* can now be located anywhere there is web access to the NCAP. Thus, we have produced a system with an effective 25000-mile long serial cable. Demonstration of the technique is illustrated in Figure 6, which shows the *HP-Vee* generated data screen during control of the viscosity monitor via the network-based serial connection. The program plots the change in frequency of the quartz resonator and the temperature at the sensor head. The data is saved to disk for further processing and analysis. The engineering trade-off for this convenience, however, is the reduction in available data rate. This limitation can often be mitigated by designing the data acquisition hardware at the sensor location such that data logging occurs over several minutes at high speed and then large packets of data are sent to the control hardware over the network. For systems that require continuous, rapid data acquisition (sample rates > 1 Hz), this system architecture will not be adequate.

One of the historic limitations of RS-232 interfacing is the number of instruments that can be addressed. Each instrument requires a single dedicated port for communication. However, this can be overcome by using a commercial automatic serial multiplexing device such as a Black Box Model COS 16[8] along with the architecture of Figure 3. This is illustrated in Figure 7 which shows a multiple-device serial architecture where one serial connection is made by the NCAP through the web to a serially addressed code-operated serial switch. The data is then directed to/from the target serial device selected by the switch.

Other Techniques

Additional NCAP Utilization

In parallel with the LDRD project work at Sandia National Laboratories, a contract was placed with Wico Information Technologies, Inc.9 (and subsequently a spin-off company, Sensor Synergy10) to further investigate NCAP utilization as a sensor network interface. The results from that contract work are summarized in the project final report11.

The system hardware consisted of the Agilent BFOOT 66501 MicroWeb server connected to a STIM through the IEEE 1451.2 interface and a Sandia liquid viscosity monitor. The Sandia viscosity monitor utilizes a 5 MHz thickness shear mode (TSM) quartz resonator along with patented Lever oscillator electronics for resonator drive and signal output (see Figure 8). Changes in liquid viscosity produce shifts in both an oscillator frequency and a damping voltage. The frequency and voltage signals

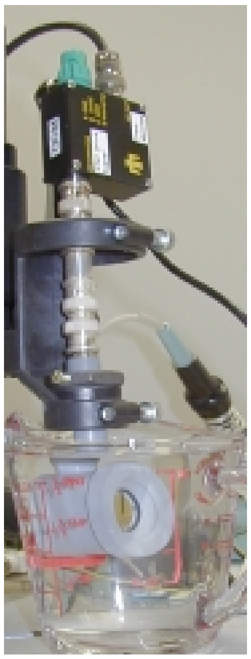


Figure 8. *Sandia TSM resonator sensor shown in a liquid bath during the viscosity testing.*

from the monitor required specially designed interface circuitry for compatibility with the STIM input. This interface circuit was built around an Analog Devices ADuC812 that includes an 8501 microcontroller core. Frequency counting was accomplished by beating the viscosity monitor signal against a fixed 5 MHz oscillator then recording the down-converted signal with one of the ADuC812 16-bit counters.

The NCAP server was loaded with a simple HTML web page containing embedded JAVA applets to interface with the STIM. Data was extracted for three channels: the resonator frequency, resonator voltage, and a separate temperature monitor placed in the liquid. Sensor signals are updated every few seconds. A display of the real-time sensor operation was provided at the URL:

http://64.220.225.186

A facsimile of the web site display is shown in Figure 10.

Wireless Internet Access

For many sensor applications at extremely remote locations, Ethernet access will not be directly available. In other situations, it will not be desirable to provide cabled network access for economic or other reasons – such as implementation on a moving platform. These cases will require some form of telemetry system to transmit sensor signals to a receiving location so that data will be available to customers. Here we discuss two systems that utilize wireless interfaces for gaining network access.

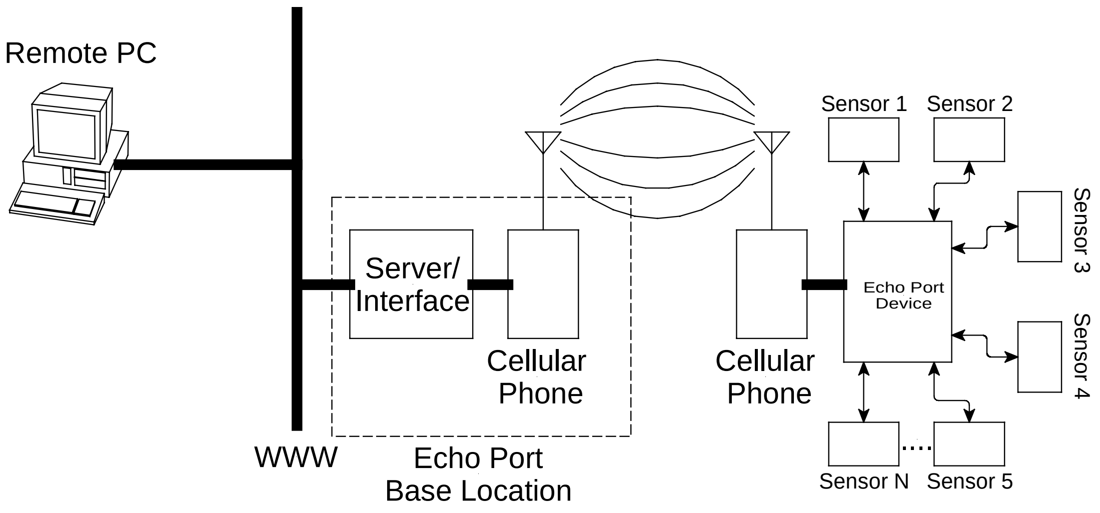
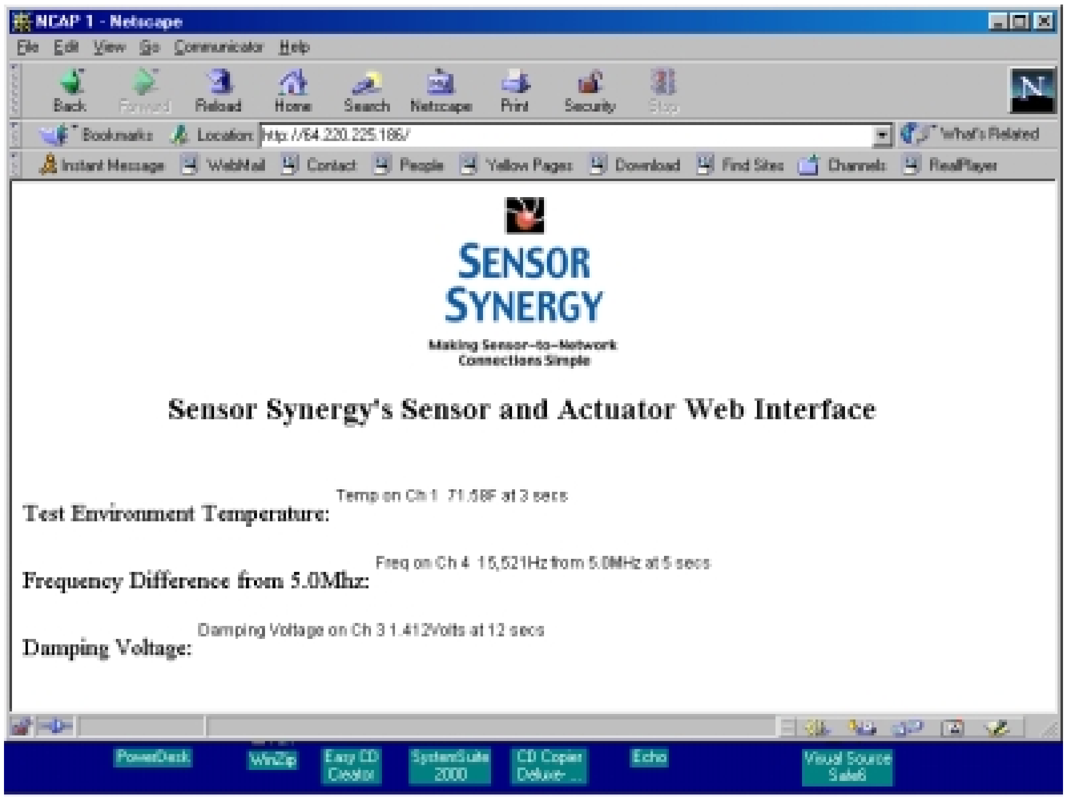


Figure 9. *Diagram of wireless communications/access from remote sensors to a base station where information/data is placed on the WWW. Echo Port is one company that builds and markets sensor devices and the remotely powered sensor interfaces using current cellular phone technology. Additionally, they provide the base location service to place information on the WWW.*

EchoPort Systems


NCAP 1 - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://64.220.225.186/
What's Related
Instant Message WebMail Contact People Yellow Pages Download Find Sites Channels RealPlayer
SENSOR SYNERGY
Making Sensor-to-Network Connections Simple
PowerDesk WinZip Easy CD Creator SystemSuite 2000 CD Copier Deluxe ... Echo Visual Source Safe6

Sensor Synergy's Sensor and Actuator Web Interface

Test Environment Temperature: Temp on Ch 1 71.58F at 3 secs

Frequency Difference from 5.0Mhz: Freq on Ch 4 15,521Hz from 5.0MHz at 5 secs

Damping Voltage: Damping Voltage on Ch 3 1.412Volts at 12 secs

Figure 10. *Display of the Sensor Synergy web page showing the Sandia viscosity monitor frequency shift and voltage along with the ambient temperature***.**

Several products and services utilizing a wireless interface are provided by EchoPort.[12] The EchoPort systems incorporate a microburst technology that uses the control channels of the AMPS cellular networks to send short packets of data. As such, service is available in all areas that have cellular coverage. Access to the WWW is provided through the EchoPort Central^{TM} base station where cellular signals are received, decoded, and made available to the customer. A typical sensor system might resemble that shown in Figure 9. In addition to placing data on the WWW, the EchoPort system also can notify customers via cell phone, pager, or through direct email messages.

Existing EchoPort systems are not designed for real-time data transfer or high-bandwidth information exchange. Typical applications target "alarm" situations in which sensors at remote locations respond to a predetermined set of conditions and signal the customer when they occur. This data exchange requires a minimal amount of cellular service. EchoPort develops, produces, and markets a number of battery or remotely powered sensors – motion sensors, periodic heartbeat monitors, video monitors, etc. Systems exist for two-way information exchange allowing some simple control operations through the EchoPort web site.
The EchoPort engineers are presently exploring new technologies that will allow higher data exchange rates through cellular channels. Such service should be available within the next two years. This higher bandwidth capability, however, will not satisfy the needs

for real-time, continuous transfer of large amounts of data. For these applications, wireless protocols other than those used with cellular telephone technology are being developed.

PNNL Remotely Attended Monitoring

Researchers at Pacific Northwest National Laboratory (PNNL) have an existing program to develop a custom system for gathering data from remote sensor instrumentation and using wireless technology to transmit the information to a base site. System design criteria call for a reusable platform for internet-aware devices, an advanced networking software architecture that allows for full hardware control, and software streaming data management. Demonstration systems are being prepared that will function with a Radio Frequency Ion Trap Mass Spectrometer (RF-ITMS) and a second-generation chemical sensor array.

The reusable embedded control network instrument is labeled Sensor Platform Architecture for Rapid Test, Analysis and Networking (SPARTAN). The instrument utilizes multi-microprocessor control for sensor interfacing, data acquisition, local data analysis and storage, and wireless networking with authentication/encryption. A real-time Linux operating system is used with the local processors to provide stability and security. The wireless interface incorporates a commercially available module from Lucent Technologies. The module operates at 2.4 GHz with 11-Mbit/sec data transmission rates and 128-bit data encryption.

System interface software is designed to operate from a browser on the WWW or run as a standalone application. The JAVA applets are sensor specific for each remote hardware application and provide for full two-way network operations. All software is designed to be platform independent. Collaboratory tools are incorporated so that many operator/users at different locations can gain remote access and participate in joint decision making.

DeviceNet Schemes

DeviceNet is a low-cost industrial network protocol for connecting hardened sensors, actuators, and operator displays to programmable logic controllers (PLCs) and PCs. Configured networks eliminate need for expensive hard-wiring while providing device-level diagnostics. The Open DeviceNet Vendor Association (ODVA)[13] is an independent organization that manages the DeviceNet technology and promotes its use in industrial automation. ODVA maintains the DeviceNet standard and ensures that product manufacturers meet stringent specifications.

The typical industrial PC or notebook computer can be connected to the DeviceNet network using commercially available controller interfaces. These interfaces utilize the standard RS-232 ports, PCI-bus cards, PCMCIA cards, etc. A number of manufacturers (over 200) sell “smart” sensors, actuators, terminals, motor controllers, and other devices that readily mate to the DeviceNet network. Manufacturers also provide device software drivers for popular programming environments such as LabVIEW, C/C++, and Visual Basic. Since all commercial components and software adhere to tight standards, system compatibility is ensured. Specialty sensors and sensor systems (such as the fluid monitors or chemical sensors) that conform to the DeviceNet protocol are not presently available, but such sensors can be custom designed and fabricated. This would allow the consumer to take advantage of the extensive networking capability already existing for the other components. Once the sensors or other industrial instruments are linked to a PC via DeviceNet, access to external networks or the WWW can be gained.

Recently, a complete network system utilizing the WWW and DeviceNet was implemented and demonstrated by Sandia National Laboratories Org. 6411 (Bruce Thompson, project lead) and the Kansas City plant. The general architecture of that system is illustrated in Figure 11. Several automated machines at the Kansas City plant were configured with DeviceNet-ready sensors that connected directly to a DeviceNet controller interface. Sensor control software was resident on an industrial-hardened PC that provided the link between DeviceNet and an internal network, and ultimately, through a server, to the WWW. The control PC also utilized Internet Messaging System software[14] that conveniently bundled sensor data and sent it to a requesting computer. Other computers on the Kansas City plant internal network were utilized for data storage, analysis, and databasing.

A significant component of the Sandia-Kansas City network demonstration was implementation of reliability analysis and predictive maintenance software developed by Org. 6411. Machine sensors installed at the Kansas City plant provided periodic performance data that could be compared to a large database of known machine parameters. This required developing custom software modules that were resident on the remote PC (in Albuquerque) that periodically queried the DeviceNet networked machine and received back packets of data. Direct IP address queries were performed through the WWW. Data manipulations and analysis were then performed by the remote PC for implementing the predictive maintenance codes.

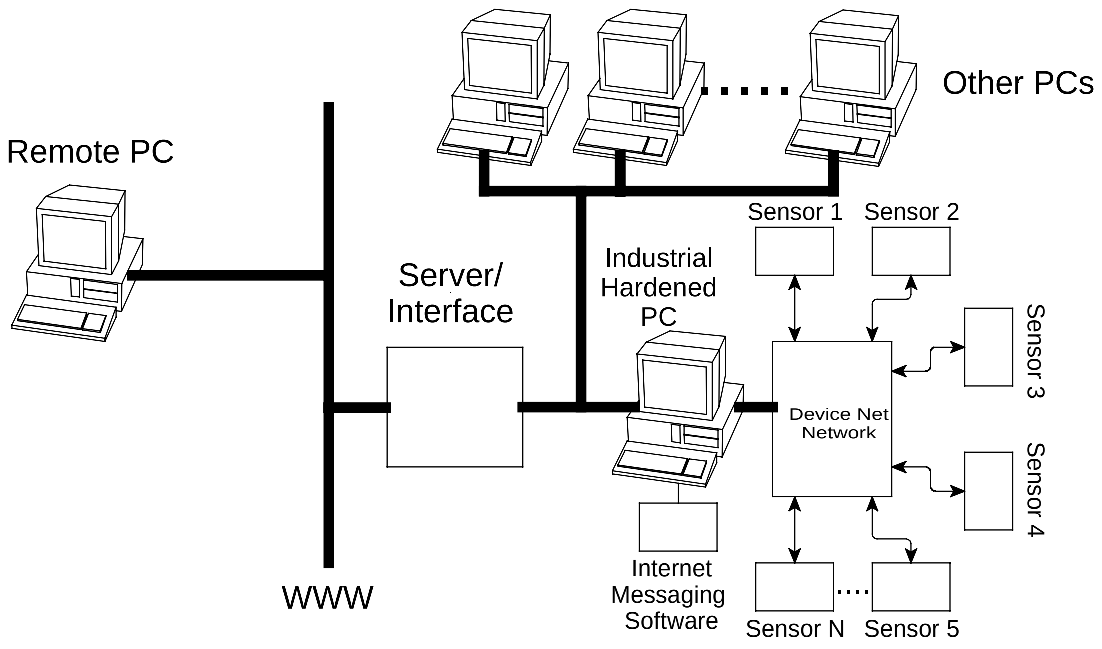


Figure 11. *Diagram of a network utilizing the Device Net Networking protocol. This type of network can be implemented as an independent network or interfaced to the WWW.*

Conclusions

We have investigated several techniques for placing sensor data on the WWW for remote interrogation by a PC. These include a simple client/server architecture, a proprietary browser approach, a serial pass-through approach, and several other approaches that have been proposed and implemented by others. It is clear that no single solution is viable for all sensing applications. One must consider several issues when designing a system for a specific application. First, data volume and sampling rate is critical, as is the ability to log data on the host PC and perform remote control of the sensor system. Second, data security issues including whether an adversary could view the data or manipulate the data must be considered. Third, division of system control between the remote hardware running autonomously and the host PC must be determined. Finally, it is important to know the remoteness of the application sensor hardware and whether the local infrastructure can support the proposed system. An appropriate system for any target application should be possible using the above approaches or variations on them.

Appendix A

HTML file that forms home page on BFOOT.

NHome.htm

```
<HTML>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
        alert("NCAP Node Wolf1")
</SCRIPT>
 </HTML>
<HTML>
<HEAD><TITLE>Laboratory Environmental Conditions</TITLE></HEAD>
<FRAMESET ROWS="25%,*">
        <FRAME SRC="NTitle.htm" SCROLLING=AUTO>
        <FRAMESET COLS="25%,*">
                <FRAMESET ROWS="25%,*">
                        <FRAME SRC="NEnviron1.htm" NAME="controls">
                        <FRAME SRC="NPix.htm">
                </FRAMESET>

                <FRAME SRC="NTemp1.htm" NAME="data">
        </FRAMESET>
</FRAMESET>
</HTML>
```

<u>Pages Called from Home Page</u>

NTitle. Htm

```
<html>
<h1 align=center>
<head><title> Sandia Cyber Sensor </title></head>
</h1>

<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
        <!-- Hide script from old browsers

        dayName = new Array
("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
        monName = new Array ("January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December")

        now = new Date

function showMilitaryTime() {
                if (document.theForm.showMilitary[0].checked) {
                        return true
                }
                return false
        }
        function showTheHours(theHour) {
                if (showMilitaryTime() || (theHour > 0 && theHour < 13)) {
                        return (theHour)
                }
                if (theHour == 0) {
                        return (12)
                }
                return (theHour-12)
        }

        function showZeroFilled(inValue) {
                if (inValue > 9) {
                        return ":" + inValue
                }
                return ":0" + inValue
        }

        function showAmPm() {
                if (showMilitaryTime()) {
                        return ("")
                }
                if (now.getHours() < 12) {
                        return (" am")
                }
                return (" pm")
        }

        function showTheTime() {
                now = new Date

                document.theForm.showTime.value = showTheHours(now.getHours()) +
showZeroFilled(now.getMinutes()) + showZeroFilled(now.getSeconds()) + showAmPm()
                setTimeout("showTheTime()",1000)
        }



        // End hiding script from old browsers -->
</SCRIPT>

<BODY onLoad="showTheTime()" BGCOLOR=CORNFLOWERBLUE TEST=BLACK LINK=RED
VLINK=PURPLE>

<h1 ALIGN=CENTER> Current Laboratory Environmental Conditions </h1>
<A HREF="http://www.mdl.sandia.gov/sensors/sensor_about.html" TARGET="_top"><IMG
SRC="R&D4.gif" ALIGN=LEFT VSPACE=5 HSPACE=5></A>
<A HREF="http://www.sandia.gov" TARGET="_top"><IMG SRC="tbird_blue.gif" ALIGN=RIGHT
VSPACE=5 HSPACE=5></A>
<hr>
<h2>
```

```
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
        <!-- Hide script from old browsers

        document.write("<H1 ALIGN=CENTER>Today is " + dayName[now.getDay()] + ", " +
monName[now.getMonth()] + " " + now.getDate() +", " + now.getFullYear() + " </H1>")


        // End hiding script from old browsers -->
</SCRIPT>
<H2 ALIGN=CENTER>

        <FORM NAME="theForm">

                <p> The current time is:
                <INPUT TYPE=TEXT NAME="showTime" SIZE=11>
                (military time?)
                <INPUT TYPE=RADIO NAME="showMilitary" CHECKED>Yes
                <INPUT TYPE=RADIO NAME="showMilitary">No
        </FORM>

</h2>


</body>
</html>
```

Nenviron1.htm

```
<html>

<BODY BGCOLOR=CORNFLOWERBLUE TEST=BLACK LINK=RED VLINK=PURPLE>



<UL>
 <LI><A HREF="/bin/1451dot2/trend" TARGET="data">Temperature</A>

 <LI><A HREF="NHumidity1.htm" TARGET="data">Humidity</A>
 <LI><A HREF="NPressure1.htm" TARGET="data">Pressure</A>
 <LI><A HREF="/bin/1451dot2/trend" TARGET="data"
        hide_ui=1 >Oxygen Concentration</A>
</UL>
<HR>
<HR>


</BODY>

</TITLE>
```

NPix.htm

```
<HTML>

<BODY BGCOLOR=CORNFLOWERBLUE TEST=BLACK LINK=RED VLINK=PURPLE>

        <IMG SRC="carney1.gif" ALIGN="CENTER">
        <IMG SRC="okane-arr.gif" ALIGN="CENTER">
        <IMG SRC="mc-refuel.gif" ALIGN="CENTER">

</BODY>



</HTML>
```

NTemp1.htm

```
<HTML>

<BODY BGCOLOR=CORNFLOWERBLUE TEST=BLACK LINK=RED VLINK=PURPLE>

<A HREF="/bin/1451dot2/trend"


</A>

 Hello! Temperature Page
</BODY>
</HTML>
```

Appendix B

Visual Basic Proprietary Browser

```
Private Sub Commandl_Click()
    ' Text2.Text = "http://169.254.126.20/bin/node/blink" 'Jamie's
     Text2.Text = "http://169.254.126.31/bin/node/blink"  'Kent's
     WebBrowser1.Navigate Text2.Text
End Sub
Private Sub Command2_Click()
     'Text2.Text = "http://169.254.126.20/bin/node/time"   'Jamie's
     Text2.Text = "http://169.254.126.31/bin/node/time"  'Kent's
     WebBrowserI.Navigate Text2.Text
End Sub
Private Sub Command3_Click()
    ' Text2.Text = "http://169.254.126.20/bin/1451dot2/summary"
     Text2.Text = "http://169.254.126.31/bin/1451dot2/summary"  'Kent's
     WebBrowserI.Navigate Text2.Text
End Sub
Private Sub Command4_Click()
```

```
 'Call ReadChannels

End Sub

 'Option Explicit
Private Sub Command5_Click()
Forml.Textl.Text = Form1.lnet1.OpenURL(Forml.Text2.Text) 'Uses Internet Transfer object

End Sub

Private Sub Command6_Click()
Load Form2
Form2.Show
 'to load nodes into Listboxl
For i = 1 To NoNodes
Form2.List2.Addltem ("Node " & Str(i) & ", " & Label (i))
Next i


End Sub
Private Sub Command7_Click()
Load.Form3
Form3.Show
Form3.Text1.Text = "1"
End Sub

Private Sub Command8_Click()
End Sub

Form3 - 1
Private Sub Command8_Click()
For i = 1 To NoNodes 'to update values from flexgrid1
     '  Form3.MSFIexGridl.TextMatrix(i,0) = i
    Label(i) = Form3.MSFIexGrid1.TextMatrix(i,1)
    URL(i) = Form3.MSFIexGrid1.TextMatrix(i,2)
    Password(i) = Form3.MSFIexGridl.TextMatrix(i,3)
    NoChannels(i) = Form3.MSFIexGridl.TextMatrix(i,4)
    Status(i) = Form3.MSFiexGridl.TextMatrix(i,5)
Next i
Dim Jmax As Integer
Load Form4
Form4.Show
Form4.MSChart1.Visible = False
Form4.Labell.Visible = False
Form4.Label2.Visible = False
Form4.Text1.Visible = False
Form4.Text2.Visible = False
For i = 1 To NoNodes ' to find maximum number of channels on any node
         If NoChannels(i) > Jmax Then Jmax = NoChanneis(i)
Next i
Form4.MSFIexGrid1.Rows = Jmax + 1
Form4.MSFIexGrid1.Cols = NoNodes + 1
Form4.MSFIexGrid1.Width =(NoNodes + 1)*1550
```

```
Form4 .MSFlexGrid1.Height = (Jmax + 1)*325
For i = 1 To NoNodes + 1
        Form4.MSFIexGridl.ColWidth(i-1) = 1400
Next i
Form4.MSFlexGrid1.TextMatrix(0, 0)= “Channel”
For i = 1 To NoNodes
Form4.MSFlexGrid1.TextMatrix(0,i)=”Node “&Str(i)
Next i
End Sub

Private Sub Form_Load()
MSFiexGrid1.Row = 1
MSFlexGrid1.Col = 1
Text2.Text = ""
Text2. Visible = False

End Sub
Private Sub MSFlexGrid1_EnterCell()
        If MSFlexGrid1.MouseRow = 0 Or MSFlexGrid1.MouseCol = 0 Then
                Text2.Visible = False
                Exit Sub
        End If
        Text2.Text = ""
        Text2.Visible = False
        Text2.Top = MSFlexGrid1.Top + MSFlexGrid1.CellTop
        Text2.Left = MSFlexGrid1.Left + MSFlexGrid1.CellLeft
        Text2.Width = MSFiexGrid1.CellWidth
        Text2.Height = MSFlexGrid1.CellHeight
        Text2.Text = MSFlexGrid1.Text
        Text2.Visible = True
        Text2.SetFocus
End Sub
Private Sub MSFIexGrid1_LeaveCell()
MSFlexGrid1.Text = Text2.Text
End Sub

Form3 - 2
Private Sub Text1_Change()
NoNodes - Val( Form3.Textl.Text)
ReDimLabel(NoNodes), URL(NoNodes), Password(NoNodes), NoChannels(NoNodes), Status(NoNodes)

If NoNodes >= 1 Then  'for default values
        URL(1) = "http://169.254.126.31"
        Password(1) = "howlhowl"
        NoChannels(l) = 7
        Status (1) = True
End If
If NoNodes >= 2 Then 'more default values
        URL(2) = "http://169.254.126.31"
        Password(2) = "howlhowl"
        NoChannels(2) = 5
        Status(2) = True
End If
'end of default definitions
```

```
Form3.Textl.Text= Str(NoNodes)
Forms.MSFlexGrid1.Rows = NoNodes + 1
Form3 MSFlexGrid1.Cols = 6
Form3 MSFlexGrid1.FormatString = “<Node No.|^Label|^URL|Password|^Channels|Status”
Form3 MSFlexGrid1.ColWidth(O) = 1200
Form3 MSFlexGrid1.ColWidth(l) = 2400
Form3 MSFlexGrid1.ColWidth(2) = 2400
Form3 MSFlexGrid1.ColWidth(3) = 2400
Form3 MSFlexGrid1.ColWidth(4) = 1200
Form3 MSFlexGrid1.ColWidth(5) = 1000
Form3 MSFlexGrid1.Height = (NoNodes +1) * 450
Form3.MSFlexGrid1.Width = 10600



For i = 1 To NoNodes
Form3.MSFlexGrid1.TextMatrix(i, 0) = i
Form3.MSFlexGrid1.TextMatrix(i, 1) = Label(i)
Form3.MSFlexGrid1.TextMatrix(i, 2 = URL(i)
Form3.MSFlexGrid1.TextMatrix(i, 3) = Password(i)
Form3.MSFiexGridI.TextMatrix(i, 4) = NoChannels(i)
Form3.MSFIpxGridI.TextMatrix(i, 5) = Status(i)
Next i

Sub End

Form 4 - 1
Private Sub Commandl_Click()
Dim i As Integer, j As Integer
For i = I To NoNodes
        Call ReadChannels(i)    'reads all channels from node i
        For j = 1 To NoChannels(i)
                If i = 1 Then Form4.MSFlexGridl.TextMatrix(j, 0) = Str(j)
                Form4.MSFIexGrid1.TextMatrix(j, i) = Readings(j)
        Next j
Next i

End Sub
Private Sub Conmmand2_Click()
Load Form5
Form5.Show

Form5.Text1.Text = "3" 'default data recording interval (sec)

Form5.Text2.Text = Str(Time()) 'default start time is now

Form5.Text3. Text = Str (Time () + 1/24)  'default end time is an hour later

Form5.Text4.Text = "Test1"

Form4.Text1.Text = "1"
Form4.Text2.Text = "2"
End Sub
Private Sub Command4_Click()
End
```

```
End Sub
Private Sub MSFlexGrid1 Click()

Form4.Text2.Text = MSFlexGridl.Row   'row selected
Form4.Text1 .Text = MSFlexGridl.Col   'column selected

 'Form4.MSChart1.Visible = False

Call PlotSelectedData(Val(Form4.Textl. Text), Val(Form4.Text2.Text), Npoints)
End Sub

Form5 - 1
Private Sub Command1_Click()
Dim Interv As Single, Tstart As Date, Tend As Date, TimeNow As Date
Dim i As Integer, j As Integer, k As Integer
 'Static k As Integer  'to store line index between calls
 'Load Form4
Form5.Hide
 ' Form4 .MSChart1.Visible = True
Form4.Textl.Visible = True
Form4.Text2.Visible = True
Form4.Label1.Visible = True
Form4.Label2.Visible = True
Call PlotSelectedData(1, 1, 0)
Interv = Val (Textl. Text) ' data recording interval
Tstart = Text2.Text ' start time for data recording
Tend = Text3.Text   'end time for data recording
 '3:  Interv = Val(Textl.Text)  'data recording interval
3:  TimeNow = Time()
If TimeNow > Tend Then GoTo 5
If TimeNow >= Tstart And Second (TimeNow - TimePrev) > Interv Then
        'Text7.Text = TimeNow
        For i = 1 To NoNodes
                    k = k + 1   ' line counter for DataArray
                    DataArray(k, 1) = i  'node number
                    DataArray (k, 2) = TimeNow  'current time
                    Call ReadChannels(i)   'gets readings from channels on node i
                          For j = 1 To NoChannels(i)
                                If i = 1 Then Form4.MSFlexGridl.TextMatrix(j, 0) = Str(j)  'to generate
                                first column of flexgrid
                                Form4.MSFlexGridl.TextMatrix (j, i) = Readings (j)    'to generate columns
                                of flexgrid
                                DataArray(k, j +2) = Readings(j)   'to store data for plotting
                          Next j

       Next i
       Npoints = k
       Call PlotSelectedData(Val(Form4.Textl.Text), Val(Form4.Text2.Text), Npoints)
     TimePrev = TimeNow ' to store time of last measurement
End If

GoTo 3:
5:  End Sub
```

```
Private Sub Command2_Click()
End
End Sub


Module1 - 1
Public DataArray(1 To 5000, 1 To 10), Readings () As Double, NoNodes As Integer, Times () As Double
Public Label 0 As String, URL() As String, Password () As String, NoChannels() As Integer, Status() As
Boolean
Public Npoints As Integer
Public Sub ReadChannels(ByVal i As Integer)  ' i is node no.
Dim Response As String
        Forml.Text2.Text = URL(i) & "/bin/1451dot2/read?startChan=0&stopChan=" & Str(NoChanneis(i))
   'Kent's
        Form1.WebBrowser1.Navigate Form1.Text2.Text
        Forml.Textl Text = ""
DoEvents
Response = Form1.Inetl.OpenURL (Form1.Text2.Text) 'Uses Internet Transfer object
Forml.Textl.Text = Response
Call ParseResponse (Response, i)  'i is node number
End Sub
Public Sub ParseResponse (ByVal Response As String, ByVal i As Integer)
Dim Tb As String, N1 As String, ChannelNo As Integer
Dim PosO As Integer, Pos1 As Integer, Pos2 As Integer, Pos3 As Integer, j As Integer
Tb = Chr(9)   ' tab delimeter between data fields
Nl = Chr(IO)  ' new line at end of data field
ReDim Readings (I To NoChannels(i))
ReDim Times (1 To NoChannels(i))
PosO = 0  ' starting position for tab search
5:  Pos1 •= InStr (PosO + 1,  Response,  Tb)  'position of first tab delimeter
ChannelNo = Val(Mid(Response, PosO + 1, Posi - PosO - 1))  'extracts channel no.
Pos2 = InStr (Pos1 + 1, Response, Tb) 'position of second tab delimeter
Readings(ChannelNo) = Val(Mid(Response, Pos1 + 1, Pos2 - Pos1 - 1)) ' extracts channel reading
Pos3 = InStr (Pos2 + 1, Response, Nl)  'position of new line delimeter
Times (i) = Val(Mid(Response, Pos2 + 1, Pos3 - Pos2 - 1)) 'extracts time number

If Pos3 < Len (Response) Then
        PosO = Pos3  'sets nl delimeter as position to start next search
        GoTo 5   'to start search for data from next channel
End If
'Form2.Text4.Text = Str(ChanelNo)
'Form2.Text5.Text = Str (Temp)
'Forni2.Text6.Text = Str(TimeNo)
'Form2.Listl.Addltem (Str(ChanelNo) & ", " & Str(Time()) & ", " & Str(Temp))
'Ntimes = Ntimes +1 'no. of time points recorded
'DataArray(Ntimes, 1) = Ntimes  '=Str (Time ())   'x-axis value
'DataArray (Ntimes, 2) =• Temp         'y axis value

End Sub
Public Sub PlotData()
Static Initialize As Integer
Static Ymax As Single, Ymin As Single
Static PlotArray(1 To 50, 1 To 2)


Module1 - 2

Dim i As Integer
If Initialize = 0 Then ‘ to initialize plot limits
 Initialize = 1
Ymax = -100000000#
Ymin = l00000000#
End If


For i = 1 To 49  'to shift PlotArray elements (to mimic chart recorder)

        PlotArray(i, 1) = PlotArray(i + 1, 1)
        PlotArray(i, 2) = PlotArray(i + 1,2)
'    If PlotArray(i, 2)  > Ymax Then Ymax = PlotArray(i,2)
'    If PlotArray(i, 2)  < Ymin Then Ymin = PlotArray(i,2)
Next i

        PlotArray(50, 1) = Str(Time()) ‘insert new valuew
        PlotArray(50, 2) = Temp ‘ChannelReadings(1) ‘insert new value
If PlotArray(50, 2) > Ymax Then Ymax = PlotArray(50,2)
If PlotArray(50, 2) < Ymin Then Ymin = PlotArray(50,2)

    NumChannels = 1 ' no. of lines being plotted

Form2.MSChart1.chartType = VtChChartType2dLine
Form2.MSChart1.RowCount = 50 'sets row number
Form2.MSChart1.ColuinnCount = NuinChannels
Form2.MSChart1.Plot.UniformAxis = False
Form2.MSChart1.Plot.Axis (VtChAxisldY2).AxisScale.Hide = True
Form2.MSChart1.Plot.Axis(VtChAxisldY).CategoryScale.Auto = False
Form2.MSChart1.Plot.Axis(VtChAxisldY).ValueScale.Minimum = Ymin
Form2.MSChart1.Plot.Axis (VtChAxisldY).ValueScale.Maximum = Ymax
Form2.MSChart1.ChartData = PlotArray

End Sub


Public Sub PlotSelectedData(ByVal m As Integer, ByVal n As Integer, ByVal kk As Integer)
Static Initialize
Dim Ymax As Single, Ymin As Single
Static PlotArray(1 To 50, 1 To 2)
Dim i As Integer, k As Integer, StartVal As Integer, StopVal As Integer
Ymax = -100000000#
Ymin = 100000000#
 'StartVal = kk - 100
 'StopVal = kk
If kk = 0 Then GoTo 8

k = kk + 1
i = 51
```

```
2:k = k-1 'to load latest 50 data points into PlotArray:
    If DataArray(k, 1) = m Then  'select data from correct node:
        i = i - l
        PlotArray (i, 1) = Str(DataArray(k, 2)) 'time data
        PlotArray (i, 2) = DataArray(k, n + 2) 'data reading for channel n
        If PlotArray(i, 2) > Ymax Then Ymax = PlotArray(i, 2)
        If PlotArray(i, 2) < Ymin Then Ymin = PlotArray(i, 2)
    End If
If i > 1 And k > 1 Then GoTo 2
8: NumChannels = 1  'no. of lines being plotted
Form4.MSChartl.Visible = True
Form4 .MSChart1.chartType = VtChChartType2dLine
Form4 .MSChart1.RowCount = 50 'sets row number
Form4.MSChart1.ColumnCount = NumChannels
Form4 .MSChart1.Plot.UniformAxis = False


Module1 - 3
Form4.MSChartl.Plot.Axis(VtChAxisIdY2).AxisScale.Hide = True
```

References:

[1] Agilent Technologies, Inc., Distributed Measurement and Control Operation, 1501 Page Mill Road, Palo Alto, CA 94304-1126..

[2] IEEE Std 1451.2-1997, *Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*, Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, 08855, Sept 26, 1997.

[3] Telemonitor, Inc., 9055F Guilford Road, Columbia, MD 21046.

[4] HY CAL Engineering, 9650 Telstar Ave, El Monte, CA 91731-3093.

[5] Netscape Communications Corp., 501 E. Middlefield Road, Mountain View, CA 94043.

[6] Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399

[7] Tactical Software, 402 Amherst St. Suite 402, Nashua, NH 03063.

[8] Black Box Network Services, 1000 Park Drive, Lawrence, PA 15055-1018.

[9.] Wico Information Technologies, Inc., 1110 Lake Cook Rd., Suite 220, Buffalo Grove, IL 60089.

[10.] Sensor Synergy, 1110 Lake Cook Rd., Suite 340, Buffalo Grove, IL 60089.

[11.] James Wiczer, "Ethernet Interface for Sandia's Thickness Shear Mode Quartz Crystal Microbalance Resonator Sensor," Final Report, Contract BG-5676, Wico Information Technologies, Inc., Sept. 18, 2000.

[12.] EchoPort, 4101 Indian School Rd NE, Suite 440, Albuquerque, NM 87110. http://www.echoport.com

[13.] Open DeviceNet Vendor Association, PMB 499, 20423 State Rd 7 #F6, Boca Raton, FL 33498-6797, http://216.10.36.18/index.htm

[14.] Synergy Software Technologies, Inc., 159 Pearl Street, Essex Junction, VT 05452.

Distribution:

1	MS 0188	LDRD office
1	9018	Central Technical Files, 8940-2
2	0899	Technical Library, 4916
1	0619	Review & Approval Desk, 12690 For DOE/OSTI
5	1425	K. B. Pfeifer, 1744
2	1425	R. W. Cernosek, 1744
2	1425	S. J. Martin, 1744
2	1425	R. W. Waldschmidt, 1744
2	1425	A. N. Rumpf, 1744

[1] Agilent Technologies, Inc., Distributed Measurement and Control Operation, 1501 Page Mill Road, Palo Alto, CA 94304-1126..

[2] IEEE Std 1451.2-1997, *Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*, Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, 08855, Sept 26, 1997.

[3] Telemonitor, Inc., 9055F Guilford Road, Columbia, MD 21046.

[4] HY CAL Engineering, 9650 Telstar Ave, El Monte, CA 91731-3093.

[5] Netscape Communications Corp., 501 E. Middlefield Road, Mountain View, CA 94043.

[6] Microsoft Corporation, One Microsoft Way, Redmond, WA 98052-6399

[7] Tactical Software, 402 Amherst St. Suite 402, Nashua, NH 03063.

[8] Black Box Network Services, 1000 Park Drive, Lawrence, PA 15055-1018.

[9.] Wico Information Technologies, Inc., 1110 Lake Cook Rd., Suite 220, Buffalo Grove, IL 60089.

[10.] Sensor Synergy, 1110 Lake Cook Rd., Suite 340, Buffalo Grove, IL 60089.

[11.] James Wiczer, "Ethernet Interface for Sandia's Thickness Shear Mode Quartz Crystal Microbalance Resonator Sensor," Final Report, Contract BG-5676, Wico Information Technologies, Inc., Sept. 18, 2000.

[12.] EchoPort, 4101 Indian School Rd NE, Suite 440, Albuquerque, NM 87110. http://www.echoport.com

[13] Open DeviceNet Vendor Association, PMB 499, 20423 State Rd 7 #F6, Boca Raton, FL 33498-6797, http://216.10.36.18/index.htm

[14] Synergy Software Technologies, Inc., 159 Pearl Street, Essex Junction, VT 05452.